Метрики за тестване на софтуер - ефективност на отстраняване на дефекти (DRE)

Най-просто DRE (Defect Removal Efficiency) може да се изрази като процент, където

$$DRE = \frac{\textit{общи дефекти , открити по време на работата върху проекта}}{\textit{общи дефекти на проекта}} . 100$$

Как да определим общия брой дефекти в рамките на решението?

Едно от най-съществените предизвикателства при изчисляването на процента DRE е определянето на общия брой дефекти от проекта в системата. Има няколко начина как това може да се определи :

1. Defect Seeding - тук дефектите се въвеждат нарочно в кода, за да се определи ефективността на програма за тестване. Defect Seeding се използва в редица академични проучвания, но рядко се използва в реални тестове, където времевите рамки и бюджетите често вече са разпределени, създаването на дефекти е трудна и отнемаща време дейност.
   The total number of defects in a application can be extrapolated to = total defects found during testing x (total defects seeded/total seeded defects detected)

2. Оценка на дефектите - Това включва оценка на броя на дефектите в системата въз основа на предишни резултати и опит в индустрията. Това е техника, която е малко вероятно да даде наистина точен брой дефекти и ще има по-голяма полза като първоначална оценка на планирането на тестовете.

3. Преброяване на дефекти - Това включва комбиниране на „броя на откритите дефекти по време на тестване" с „броя на дефектите, идентифицирани по време на производство, проследявани обратно до проекта". Този метод винаги ще даде по-ниска стойност от истинския брой дефекти, въведени от проекта като -
   - Не всички дефекти се проявяват като неуспехи в производството
   - Свързаните с използваемостта дефекти често са по-малко вероятно да бъдат повдигнати в произвеждането.
   - За да бъдат полезни DRE метриките, те трябва да бъдат извлечени възможно най-близо до момента, в който системата излезе от теста, но много функции може да не се изпълняват в представянето и произвеждането, докато системата не е активирана в продължение на няколко години.

Това, което бих препоръчала в повечето случаи, е използването на „Defect Counting" и намаляване на броя на производствените дефекти, след като системата е била активирана в продължение на три месеца. Това трябва да е достатъчно, за да бъдат идентифицирани повечето важни проблеми, като същевременно все още предоставя информация в подходящ срок.

Нормализиране на данните за дефектите

За да бъдем ефективни, е важно дефектите да бъдат постоянно повдигани под въпрос и класифицирани през целия тестващ жизнен цикъл и производство, това означава:

- Ако е необходимо, прекласифицирайте приоритетите на производствения дефект, за да се гарантира, че те съответстват на дефектите, открити в теста

- Анализирайте всички заявки за промяна на производството, за да видите дали те действително адресират дефекти
- Често може да има забавяне с намирането на производствените дефекти, така че е важно да се говори с някои от основните потребители на системата, за да се идентифицират всички проблеми, които са наблюдавали, но все още не са официално регистрирани

Кога са въведени дефектите?

За да се измери ефективността на конкретни тестови екипи или тестови фази, е необходимо да се определи кога и къде в рамките на проекта са въведени дефекти, това изисква анализ на нивото на първопричината за вероятната причина за всеки дефект. Дефектите обикновено се класифицират като въведени в следните области –

- Изисквания
- Дизайн
- Изграждане
- Внедряване в производство

За итеративен проект също е добра практика да се запише итерацията, в която са въведени дефектите.

Примерна формула

Phase Specific DRE

Това измерва колко ефективна е тестовата фаза за идентифициране на дефекти, които са предназначени за улавяне

1. DRE Requirements Inspection = (number of requirements related defects identified during requirements inspection/total number of requirements defects identified within the solution)
2. DRE Design Inspection = (number of design and requirement related defects identified during design inspection/total number of design and requirement defects identified within the solution)
3. DRE Unit Test = (number of unit test defects identified during unit test/total number of unit test defects identified within the solution)*
4. DRE Integration Test = (number of integration defects identified during integration test/ (total number of integration test defects identified within the solution post-unit test)
5. DRE System Test = (number of system test defects identified during system test)/(total number of system test defects identified within the solution post-Integration test)
6. DRE Acceptance Test = (number of acceptance test defects identified during acceptance test)/(total number of acceptance test defects identified within the solution post system test)

Overall DRE

Това измерва колко ефективна е тестовата фаза за улавяне на всички остатъчни дефекти в приложението, независимо от фазата, която би трябвало да ги залови. (Като пример Acceptance Testing не се опитва конкретно да открие дефектите на Unit Test, но цялостната програма за тестване ще обхване много пътища чрез функционалността и трябва да идентифицира пропуснати дефекти от други фази).

1. Overall DRE System Test = (number of defects identified during system test)/(total number of functional defects identified within the solution post-Integration test)
2. Overall DRE Acceptance Test = (number of defects identified during acceptance test)/ (total number of functional defects identified within the solution post-system test)

Какво е добър DRE резултат?

Средният DRE резултат обикновено е около 85% при пълна програма за тестване, но при задълбочени и всеобхватни изисквания и процес на проверка на дизайна това може да се очаква да се повиши до около 95%.

Източници:
https://www.equinox.co.nz/blog

Коментар:

Това според мен е една доста радикална и интересна метрика. Да предположим, например, че по време на етапа на качествения контрол или тестване са открити 100 дефекта и 90 от тези дефекти са отстранени от екипа за разработка още в момента на измерването. Тогава DRE ще бъде изчислено като 90, разделено на общия брой, което ще доведе до 90%. Сега възниква въпроса: какъв е добрия DRE резултат? Метриката, която колежката така добре е описала, сочи че това би било добър резултат. Очевидно е, че колкото повече дефекти се отчитат при тестването и по-малко при работа на живо, толкова по-висок е процентът и по-високо качество (уж). Перфектното приложение няма да има дефекти на живо и следователно DRE резултат от 100% (т.е. всички дефекти бяха открити при тестване). Темата е описана доста добре и цялостно, но нека проява малко скептицизъм към самата метрика. DRE се основава на оглед на разработката, тестването и качеството на софтуера. Ще дам няколко причини, поради които мисля, че DRE не би била особено валидна метрика:

- Разработката на софтуер не е предвидима, последователна производствена дейност. DRE мълчаливо предполага, че развитието е като производство, че е предвидимо упражнение за изграждане на добре разбран и определен артефакт. На всеки етап от процеса дефектите трябва постепенно да се елиминират, докато обектът не бъде завършен и DRE трябва да достигне 95% (или каквото и да е друго), че ние да не произвеждаме коли?
- Качеството според мен не е отсъствие на нещо.
- Дефектите не са взаимозаменяеми обекти. Дефектът не е обект. Той няма качества, освен тези, които решаваме да му дадем при конкретни обстоятелства. Дефектът може да е просто нещо, което не разбираме, че изисква разследване. Може да е проблем с приложението, или **може да е някаква характеристика на реалния свят**, за която не сме знаели и която ще изисква от нас да променим приложението, за да се справим. Следователно какъв ще е смисъла да броим дефекти, които не разбираме или не знем?
- DRE има опростена стандартизирана представа за времето. Прилагането на 90-дневно прекъсване за изчисляване на DRE и използването на това като мярка за качество може би не би могло да се приложи във всеки един софтуер?

Според мен DRE се опитва да предефинира реалността, за да можем да я преброим.